HÖGSKOLAN I HALMSTAD

Tentamen i programmering DI2006 2020-08-10

Tid: 9:00-12:00

Lärare: Johan Thunberg, 035-167273

Anvisningar

- **Tentan skickas in senast klockan 12:00 på blackboard under fliken ”tenta”. Inskickningsmöjligheten stänger klockan 12:00. Ni kan skicka in hur många gånger som helst och sista försöket är det som räknas. Inga undantag görs för sen inlämning, så skicka in i tid!**
- Svaren sparas helst i en .py-fil där svaren på frågorna ges som kommentarer och svar i form av kod är i körbart skick. En vanlig text-fil (dvs .txt) går också bra. En pdf-fil är ok, men se i sådana fall till att det inte uppstår formateringsproblem vid konvertingen till pdf. Examinator/rättande lärare bedömer endast det som har skickats in.
- Det är tillåtet att använda en Pythoninstallation, men det är inte tillåtet att kopiera lösningar från andra studenter eller internet. Det är heller inte tillåtet att på något sätt samarbeta med andra studenter (eller individer). Svaren kommer att undersökas för plagiering.
- Vid varje uppgift anges hur många poäng som maximalt utdelas för uppgiften.
- Denna tentamen består av två delar: **Del 1** och **Del 2**. På Del1 kan man få maximalt 30 poäng. På Del 2 kan man få maximalt 20 poäng.
- För betyg 3 gäller det att få minst 15 poäng på Del 1 (Del 2 behöver alltså inte göras för att få detta betyg).
- För betyg 4 gäller det att
    - få minst 15 poäng på Del1 och minst 8 poäng på Del 2,
    - samt muntligen examineras senare idag (Tenta 2) på frågorna i Del 2 samt relaterade frågor, varefter en samlad bedömning avgör betyget.
- För betyg 5 gäller det att

- få minst 15 poäng på Del1 och minst 13 poäng på Del 2,
- samt muntligen examineras senare idag (Tenta 2) på frågorna i Del 2 samt relaterade frågor varefter en samlad bedömning avgör betyget.

- För att få göra den muntliga tentamen för högre betyg så måste man vara anmäld till den. Om man ändrar sig och inte längre är intresserad av ett högre betyg så avstår man från att ansluta till den muntliga redovisningen. Anslut i tid om i vill examineras muntligt. Om ni ansluter till mötet sent, finns mindre tid för att göra en bedömning.

Del 1 (Denna del består av sex uppgifter)

Uppgift 1 (6p)

a) Betrakta programmet nedan. Syftet med programmet är att användaren anger en summa pengar i form av ett positivt heltal. Sedan räknar programmet ut hur många hundralappar, femtiolappar, tjugolappar, och enkronor denna summa motsvarar och skriver ut detta på skärmen. Programmet fungerar emellertid inte som det ska. Programmeraren har glömt att (skapa och) tilldela värden till de fyra variablerna `antal_hundralappar`, `antal_femtiolappar`, `antal_tjugolappar` och `antal_enkronor`. Er uppgift är att göra detta på raderna 3, 5, 7 och 9 så att programmet fungerar som det är tänkt. Svara på följande sätt:
rad 3: …
rad 5: …
rad 7: …
rad 9: …

```
1 pengar = int(input("Ange hur mycket pengar:"))
2 rest = pengar
3
4 rest = rest%100
5
6 rest = rest%50
7
8 rest = rest%20
9
10 print(pengar, "kronor motsvarar", antal_hundralappar,
11       "hundralappar,", antal_femtiolappar,
12       "femtiolappar,", antal_tjugolappar,
13       "tjugolappar och", antal_enkronor, "enkronor.")
```

b) Betrakta följande uttryck: `((4 A 2) or (1.0 B 1.0))` där **A** och **B** ska bytas ut mot relationsoperatorer (t.ex. ==, <, >) sådana att uttrycket blir `True`. Uppgiften är att ange alla val av sådana relationsoperatorer **A** och **B** så att uttrycket blir `True`.

Uppgift 2 (5p)

a) Betrakta följande program:

```
x = int(input("Ange x:"))
y = 0
for i in range(x**2):
    y += i
print(y)
```

Skriv ett nytt program som fungerar på samma sätt som detta, men inte använder sig av for-loop(ar). Utskriften på skärmen ska bli densamma för det nya programmet som för det gamla programmet för samma val av input.

b) Betrakta följande program där användaren anger ett positivt heltal:

```
tal = int(input("Ange tal:"))
x,y,z = 1,1,1
while x <= tal:
    while y <= tal:
        while z <= tal:
            if x <= y and y <= z:
                if x**2 + y**2 == z**2:
                    print("(",x,",",y,",",z,")")
            z += 1
        y += 1
        z  = 1
    x += 1
    y  = 1
```

Skriv ett nytt program som fungerar på samma sätt som detta, men inte använder sig av while-loop(ar). Utskriften på skärmen ska bli densamma för det nya programmet som den för det gamla programmet för samma val av input.

Uppgift 3 (7p)

a) Betrakta funktionen `foo_bar` nedan där en del av koden har dolts (med en svart ruta innehållandes ett frågetecken). Ersätt den dolda delen med valfri kod (kan vara godyckligt antal rader med kod) så att:

`foo_bar("lol")`, returnerar strängen `"llollo"`,
`foo_bar("hej")`, returnerar strängen `"hhejje"`,
`foo_bar("hopp")`, returnerar strängen `"hhoppppo"`.

Obs! det finns såklart flera lösningar på detta problem. Alla lösningar godtages så länge funktionen returnerar enligt ovan specifikation.

```
def foo_bar(string):
    string2 = ""
    for i in range(len(string)):
        ?
    return string2
```

b) Betrakta funktionerna `bar` och `foo` nedan.

```
def bar(value):
    return value*(value-1)

def foo(value):
    # ...
    # ...
    # ...
    while True:
        if value > 10:
            break
        else:
            value += bar(value)
    return(value)
```

Antag att vi anropar `foo` med heltalsvariabeln `var` som argument, dvs. `foo(var)`. Svara nu på följande delfrågor. För varje delfråga måste ni kort motivera svaret.

i) Vad blir `x = foo(var)` om `var > 10` ?
ii) För vilka heltal `var` gäller det att `foo(var)` returnerar ett värde och inte fastnar i en oändlig loop?
iii) Vad händer om `var` är ett negativt heltal vid anropet `foo(var)`?
iv) Ersätt raderna med "# ..." med kod så att `foo` returnerar ett värde för de fall där den nuvarande implementationen av `foo` inte returnerar ett värde.

Uppgift 4 (5p)

Betrakta följande program som innehåller funktionerna `foo` och `bar`.

```
def foo(arg1, arg2):
    f = open(arg1,"w")
    for i in range(arg2):
        for j in range(i):
            f.write("1")
        f.write("\n")
    f.close()

def bar(arg1, arg2):
    foo(arg1, arg2)
    count = 0
    f = open(arg1,"r")
    for line in f:
        count += len(line) - 1 #?
    f.close()
    return count

print(bar("test", 4))
```

a) Vad gör funktionerna `foo` och `bar`? Beskriv övergripande och kortfattat.
b) Vad händer om vi tar bort ”`- 1`” på raden som avslutas med ”`#?`”
c) Vad skrivs ut på skärmen när programmet körs?

Uppgift 5 (4p)

Betrakta följande program:

```
def foo(string1, string2):
    var = {}
    for i in range(len(string1)):
        if string1[i] not in var.keys():
            if string1[i] in string2:
                var[string1[i]] = 0
                antal_forekomster = 0
                for j in string1 + string2:
                    if j == string1[i]:
                        antal_forekomster += 1
                var[string1[i]] = antal_forekomster
    return var

def bar(string1, string2):
    var1 = set(string1)
    var2 = set(string2)
    var3 = var1 & var2
    var = {}
    for i in var3:
        var[i] = 0
        antal_forekomster = 0
        for j in string1 + string2:
            if i == j:
                antal_forekomster += 1
        var[i] = antal_forekomster
    return var


print(foo("hej du", "hej hej hejsan"))
print(bar("hej du", "hej hej hejsan"))
```

a) Ange (inte rad för rad utan övergripande) vad funktionerna `foo` och `bar` gör.
b) Vad skrivs ut på skärmen när programmet körs?

Uppgift 6 (3p)

Betrakta exemplet nedan där en klass återfinns som representerar bollar (det är valfritt om ni vill att dessa ska vara tvådimensionella eller tredimensionella). Vissa delar har dolts. Ersätt de dolda delarna A, B och C med kod så att när programmet körs så skrivs volymen ut för en boll med radie 1.

```
import math

class Ball:
    def __init__(self, radius):
        [A]
    def compute_volume(self):
        return [B]

    def __str__(self):
        return "Bollen har volymen: " [C]

boll = Ball(1)
print(boll)
```

Del 2 (denna del består av två uppgifter)

Denna del göres endast om ni har för avsikt att delta i den muntliga tentamen.

Uppgift 7

a) **(5p)** Skapa en klass KeyValue som innehåller två variabler. Den första är en sträng som representerar en nyckel (key) den andra är ett heltal som representerar ett värde (value). När man skapar en instans (objekt) av klassen så ska man kunna ange vad dessa två variabler är. Klassen ska även innehålla följande instansmetoder:
   a. en metod som används för att ändra värdet på value-variabeln,
   b. en metod som används för att ändra värdet på key-variabeln,
   c. en metod som returnerar value-variabeln,
   d. en metod som returnerar key-variabeln,
   e. en metod som konverterar objekt av klassen till en sträng.

b) **(7p)** Programmera en metod

```
generate_map(String text)
```

som hittar alla unika tecken i strängen `text` och associerar varje sådant tecken med ett unikt (slumpmässigt valt) tal med hjälp av KeyValue, se a). Det som returneras av metoden är en lista med KeyValue-objekt. Listan är lika lång som antalet unika tecken in text. Varje KeyValue-objekt i listan har en nyckel som är ett tecken och ett värde som är ett tal. Alla tal i KeyValue-objekten i listan ska innehålla samma antal siffror. Varje tal är unikt, dvs. det är ett värde i endast ett av KeyValue-objekten.

Talen 101, 943, 301, 402 har samma antal siffror, men talen 1, 98, 101, 77 har inte samma antal siffor. Om `text = "hej hej"` så finns det fyra unika tecken i `text` (inkluderat mellanslaget). I detta fall ska således listan med KeyValue-objekt innehålla fyra objekt (ha längd fyra) och värdena för dessa objekt ska vara slumpmässigt valda unika tal med samma antal siffror.

Uppgift 8 (8p)

I denna uppgift ska ni skapa en klass Kortlek som representerar en kortlek. Det är den vanligt förekommande kortleken med 52 spelkort (4 färger och 13 olika valörer). Kraven på denna klass är att följande funktionalitet ska ingå:

- Det ska finnas en metod som drar ett slumpmässigt kort från kortleken. Om det inte finns några kort i kortleken så ska detta hanteras på lämpligt sätt.
- Det ska finnas en metod för att lägga till ett kort till kortleken. Om kortet redan finns så ska detta hanteras. Om det man försöker lägga till inte är ett kort så ska även detta hanteras.
- Det ska finnas en metod som samlar in alla kort i kortleken (så att kortleken innehåller alla 52 kort).
- Metoden __str__ ska implementeras.
- Det ska dessutom finnas en metod i klassen som jämför två kort med två andra kort och avgör vilka två kort som är bäst, eller om de är lika bra.
    - Om de första två korten är ett par (t.ex. par i sjuor), medan de andra två korten inte är ett par (t.ex. en knekt och en nia) så är det första två korten bäst.
    - Om de första två korten är ett par och de andra två korten är ett par och de första två korten har en annan valör än de andra två korten så är de två kort bäst som har högst valör. T.ex. i fallet (knekt, knekt) och (dam, dam), så är (dam, dam) bäst.
    - Om de första två korten inte utgör ett par och de andra två korten inte heller utgör ett par och det finns endast ett kort av alla fyra kort som har högst valör, så är de två kort bäst som innehåller kortet med högst valör. T.ex om de först korten är (knekt, sjua) och de andra två korten är (dam, tvåa) så är de sista två korten bäst eftersom ”dam” är det högsta totalt sätt.
    - Antag att det kortet med högst valör bland de första två korten har samma valör som det kort med högst valör bland de andra två korten och det kortet med lägst valör bland de första två korten inte har samma valör som det kort med lägst valör bland de andra två korten. I detta fall jämför man det kort med näst högst valör för de första två korten med det kort med näst högst valör för de andra två korten. Det kort med högst valör av dessa två är bäst och således är de två korten bäst där detta kort ingår.
    - Om inget av ovanstående fall gäller så är de första två korten lika bra som de andra två korten.